Name: _____

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1. (*10 points*) Write a function that takes as parameters the pointers of two integer arrays and the two array sizes, and returns the pointer of a new integer array that is the concatenation of the two input arrays.

2. (*25 points*) Create a Card class that stores as member variables a face value and a suit. The suits are hearts, diamonds, spaced, and clubs. The face values are, A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K. The member functions are as follows,

   - Default Constructor: Sets the card to the Ace of Spades.
   - Constructor: Accepts the face value and suit of the card.
   - displayCard: prints the card out to the screen in the format of 3H, AD, 5C, 10S, ...
   - equal: Returns true if the two cards are the same, false otherwise.
   - greater: Returns true if the calling object has a greater face value than the parameter card, false otherwise.

3. (*20 points*) Say we have a class structure named IntegerList which stores a dynamic array of ints called list and the number of elements in the list named numElements. So the start of the class structure is below.

   (a) Write an overloaded minus ($-$) operator that takes the difference of the two lists. That is, all the elements of the first list that are not in the second list. For example, if the first list is $[1, 7, 2, 9]$ and the second list is $[9, 3, 10, 7, 4, 16]$ the result is the list $[1, 2]$.

   (b) Write an overloaded minus ($*$) operator that returns the intersection of the two lists. That is, all the elements that are in both lists. For example, if the first list is $[1, 7, 2, 9]$ and the second list is $[9, 3, 10, 7, 4, 16]$ the result is the list $[7, 9]$.

```
class IntegerList {
    private:
        int *list;
        int numElements;
```

4. (*10 points*) Write the first line of the declaration for a SoundSystem class. Use multiple inheritance to base the class on the CDplayer class, the Tuner class, and the CassettePlayer class. Use public base class access in all cases.

5. (*15 points*) Write a templated list class that stores a pointer to an array of the templated type and an integer that stores the size of the array. Include a default constructor that will set the size to 10 and allocate the needed space for the array. Also include a copy constructor, destructor, and overloaded `[]` operator.

6. (*40 points*) Create a three classes Shape, Triangle, and Rectangle that satisfy the following.

   (a) Shape class stores the name of the object as a string (defaulted to "Shape"). Has a default constructor and one that loads the name. It also has a function draw that prints out the name of the object.

   (b) Triangle inherits off of shape. In addition to the name (which should be "Triangle") it stores the lengths of all three sides of the triangle. There is to be a default constructor that loads 1 for all three sides and one that loads in the sides as parameters. This class is also to have a draw function that prints to the screen the name of the object along with the lengths of the three sides.

   (c) Rectangle inherits off of shape. In addition to the name (which should be "Rectangle") it stores the lengths of the height and width of the rectangle. There is to be a default constructor that loads 1 for both sides and one that loads in the sides as parameters. This class is also to have a draw function that prints to the screen the name of the object along with the lengths of the height and width.

   Once that is done, create a declaration (as if in the main) of an STL vector that can store any of these types and will polymorph the draw function. That is, if we loop through the vector, calling draw on each entry, the correct draw will be called for each entry in the vector regardless of the type it is. Also, create this draw loop.

Shape Class

Triangle Class

Rectangle Class

Main Code Fragments

7. (*20 points*) Say we have a templated class structure named ListCollection that has a ListNode inner class defined as follows.

```
class ListNode {
public:
    T value;
    ListNode *next;

    ListNode(T nodeValue) {
        value = nodeValue;
        next = nullptr;
    }
};
```

The ListCollection is a linked list with a pointer head pointing to a ListNode and is defaulted to nullptr. Write the following two member functions and assume that the class structure has no other functions.

- pushBack: Takes in the value of the new node to be inserted at the end of the list and inserts a new node at the end.
- sublist: That takes in a beginning and ending position and returns a new list that consists of the nodes between the beginning and the end values inclusively. Make sure that you do error checking on the beginning and ending values.
- count: This returns the number of elements in the linked list. This function is to be recursive, with no use of loops.

8. (*15 points*) Given a class named IntQueue that stores a static array of integers named queueArray and its size named queueSize and the number of elements in the queue named numItems, write enqueue and dequeue functions that use the queue as a circular array.

9. (*10 points*) Write a recursive function to calculate the combination $\binom{n}{k}$. In mathematical terms,

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

If n or k is 0 then the value is 1, if $n = k$ the value is 1, and we will define anything outside these ranges the value is 0.

10. (*10 points each*) Write the following functions.

    (a) A templated recursive binary search of an array, you may assume the array is already sorted.

    (b) A templated selection sort of an array.

11. (*10 points*) State the definitions of Big-O, Big-Ω, and Big-Θ.

12. (*10 points*) Fill out the time complexity table below.

Algorithm	**Best**	**Average**	**Worst**
Bubble Sort			
Insertion Sort			
Selection Sort			
Quick Sort			
Merge Sort			
Linear Search			
Binary Search			